TEMPO Spring School: Theory and Numerics for Nonlinear Model Predictive Control

Exercise 2: Direct multiple shooting

J. Andersson M. Diehl J. Gillis J. Rawlings M. Zanon

University of Freiburg, March 26, 2015

Direct multiple shooting

Consider the following optimal control problem (OCP) with two states (x_0, x_1) and one control (u):

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \int_0^T x_0(t)^2 + x_1(t)^2 + u(t)^2 \, dt \\ \text{subject to} \quad & \dot{x}_0(t) = (1 - x_1(t)^2)\, x_0(t) - x_1(t) + u(t), \qquad x_0(0) = 0, \\ & \dot{x}_1(t) = x_0(t), \qquad\qquad\qquad\qquad\quad x_1(0) = 1, \\ & -1 \le u(t) \le 1, \quad \text{for } t \in [0, T], \end{aligned} \tag{1}$$

with $T = 10$.

Let us introduce a time-grid $0 = t_0 < t_1 < \ldots < t_N = T$. In the *direct multiple shooting* method, the continuous-time problem (1) is discretized in order to obtain a nonlinear program (NLP). In order to perform the discretization, the control is parametrized using basis functions with local support. The discretized state trajectory is then obtained by simulating the system dynamics separately on each interval. The integral term of the objective function is computed using a quadrature formula.

In the simplest setting, we use an equidistant time grid, i.e. $t_{k+1} - t_k = h = T/N$. We choose a piecewise constant control parametrization, i.e. $u(t) = U_k$ for $t \in [t_k, t_{k+1}], \quad k = 0, \ldots, N-1$ and define the states at each shooting node $X_k := x(t_k)$ for $k = 0, \ldots, N$. We also approximate the integral term in the objective function by a sum of the integrand evaluated at the shooting nodes t_k. This results in the NLP:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & J(w) := \sum_{k=0}^{N} \|X_k\|_2^2 + \sum_{k=0}^{N-1} U_k^2 \\ \text{subject to} \quad & X_0 = [0, 1]^\intercal, \\ & X_{k+1} = F(X_k, U_k, h), \quad -1 \le U_k \le 1, \qquad k = 0, \ldots, N-1, \end{aligned} \tag{2}$$

where $w := [X_0, U_0, \ldots, U_{N-1}, X_N]^\intercal$ and $F : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^2, \quad (x(0), u, h) \mapsto x(h)$ is the discrete-time dynamics, obtained by solving an initial-value problem over an interval of length h. Note that we have included the initial state (X_0) as a degree of freedom in (2). As explained in the lecture, this often makes sense even if the initial-value is fixed and known, as here.

Take a moment to think about how the solution of (2) relates to the solution of (1).

Tasks:

2.1 Implement a CasADi `SXFunction` $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ which takes as argument the state x and input u and returns the ODE right-hand-side $\dot{x}$. Evaluate numerically and inspect the result.

2.2 Discretize the continuous-time dynamics by using the following CasADi function

```
F = simpleRK(f, 10)
```

This implements the popular fixed-stepsize explicit Runge Kutta integration scheme of order 4 (RK4) using 10 integration steps (cf. also task 2.5 below). Evaluate it numerically and inspect the result.

2.3 Using (2), formulate the multiple single shooting NLP as an `MXFunction`. Use $N = 20$. NLP (2) has a total of $n_w = 2\,(N+1)+N$ degrees of freedom. Start by defining a variable $w \in \mathbb{R}^{n_w}$:

```
nw = 2*(N+1) + N
w = MX.sym("w",nw)
```

To get the state X_k and control U_k you can use:

```
X = [w[3*k : 3*k+2] for k in range(N+1)]
U = [w[3*k+2] for k in range(N)]
```

Solve the NLP with IPOPT as in previous exercises.

Tip: The following code could be useful for formulating the NLP:

```
[X_next] = F([X[k], U[k], h])
g.append(X_next - X[k+1])
```

2.4 Add the path constraint $x_0(t) \geq -0.25$in the continuous-time OCP (1). In direct multiple shooting path constraints are only enforced in a finite number of time points. In the simplest (and most common) setting, the path constraints are only enforced at the shooting nodes t_k. Modify your script to solve this modified problem.

2.5 **Extra**: As mentioned above, RK4 is a popular integrator scheme for optimal control (of non-stiff systems). For M time steps, the scheme is given by the following pseudocode:

input x, u, h
$\Delta t = h/M$
for $j = 1, \ldots, M$ **do**
  $k_1 := f(x, u)$
  $k_2 := f(x + \frac{1}{2}\Delta t\, k_1, u)$
  $k_3 := f(x + \frac{1}{2}\Delta t\, k_2, u)$
  $k_4 := f(x + \Delta t\, k_3, u)$
  $x := x + \Delta t\, \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
end for
return x

Implement this as an `MXFunction` in CasADi and make sure that the result agrees with what you got with `simpleRK` above when you use the same number of integration steps.